DIANA and selected applications*

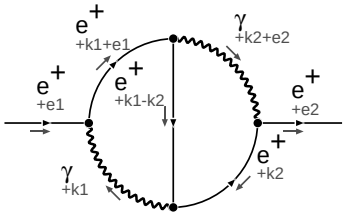
J. Fleischer, M. Tentyukov†‡ and O.V. Tarasov†‡

Universität Bielefeld, Fakultät für Physik, Bielefeld, Germany

New developments concerning the extension of the Feynman diagram analyzer DIANA are presented. We discuss new graphics facilities, different approaches to automation of momenta distribution and parallel processing facilities. Furthermore applications to $t\bar{t}$ production and Bhabha scattering are shortly discussed.

The project called DIANA (DIagram ANAlyzer)[1] for the evaluation of Feynman diagrams was started by our group some time ago. It was already used to calculate several processes [2]. The recent development[1] of this project is documented in this contribution.

The **pictorial representation** of diagrams described in [3] includes three different kinds of postscript files. Now one more kind is available, the encapsulated postscript file containing particle lines together with momenta flow [4][2]:



To use it, the user invokes the function `\outInfoEPS(Fname,Height,Font,Fontsize)`. The picture will be saved in the file `Fname`. If `Fontsize` = 0 the particle images (e.g. $e^{\pm}$) will not be printed. The width of the diagram will be calculated from "`Height`". The diagram will be scaled to fit the EPS bounding box [0 0 `Width Height`].

Some parameters may be adjusted by the user. There are two functions, `\setIEPSshift()` and `\setIEPSpars()`, which may be invoked in the "initialization" section to reset default parameters, see
http://www.physik.uni-bielefeld.de/~tentukov/printing.html#OUTINFOEPS

Particle identifiers can now be depicted by different fonts, sizes and colours. By default the particle image will be printed as in the corresponding entry of the propagator specified in the 'model file'. The user can set other images by means of the model extension, added in version 2.28, or by using the function `\setparticleimage(A,image)`. The particle images are produced by specific constructions, e.g.: `W{y(10)+}` → W^+.

'Blocks' allow local shifting etc. of certain characters. A block is started by '{' followed by a keyword with parameters in parenthesis:

`{x(#) ... }` - paints the content shifted along the abscissa (x). After the block, the current point is set to (old x, new y);

`{y(#) ... }` - paints the content shifted along the ordinate (y). After the block, the current point is set to (new x, old y);

`{xy(#)(#) ... }` - paints the content with shifts along both x and y. After the block, the current point is set to (old x, old y);

`{f(fontname)(#) ... }` - sets font `fontname` scaled by # (in fractions of 1/10 of the current font size);

`{s(#) ... }` - scales the current font by # (in fractions of 1/10 of the current font size);

`{c(#)(#)(#) ... }` - sets RGB[3] colour. Each parameter must be a positive decimal number in the region [0...1].

All sizes are in fractions of 1/10 of the base font size, i.e. "10" means the size of the current font,

*Supported by EU project HPRN-CT-2000-00149
†Supported by DFG under the project FL 241/4-2
‡On leave from JINR, Dubna, Russia
[1]For details look at
http://www.physik.uni-bielefeld.de/~tentukov/diana.html
[2]For details see also http://www.physik.uni-bielefeld.de/~tentukov/printing.html
[3]Red-Green-Blue – one of a standard colour model.

"20" is twice bigger, "5" is half size.

As `fontname` it is recommended to use only the standard 13 PostScript level 1 fonts:

Courier	*Courier-Oblique*
Courier-Bold	***Courier-BoldOblique***
Helvetica	*Helvetica-Oblique*
Helvetica-Bold	***Helvetica-BoldOblique***
Times-Roman	*Times-Italic*
Times-Bold	***Times-BoldItalic***

Σψμβολ (Symbol)

To set the image in the model, the user must add, after a line type, the character ';' and the particle image. Example:

```
[Wm,Wp;W;VP(num,ind:2,ind:1,vec,2);mmW;
        arrowWavy,3,2;W{y(10)-}]
```

In this case the line will be of a photon-like arrowed curve of width 2 and amplitude 3, and the particle image will be W^-.

As alternative, the function

`\setparticleimage(particle,image)`

overwrites the image definition coming from the model.

Topologies are represented in terms of ordered pairs of numbers like `(fromvertex, tovertex)`[1]. All external legs have negative numbers.

Often the number of topologies is too large such that it is impossible to **assign momenta** to the lines in all topologies "by hand". In that case, without special action, momenta are introduced automatically in an arbitrary manner (of course taking care of momenta conservation). The user names the loop momenta via the macro `\loopmomenta` in the "create" file, e.g. `\loopmomenta(k1,k2,k3)`, and DIANA will assign momenta automatically using "$k1$", "$k2$" and "$k3$" as the loop integration momenta.

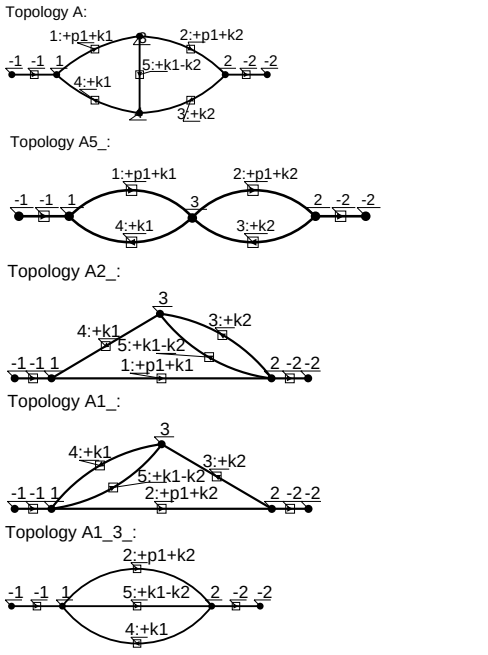
Sometimes it is important to keep some definite lines free from external momenta. If the users specifies `SET _MARK_LOOP=YES` in the "create" file, the topology editor will be invoked in a special mode, and the user "clicks" (by mouse) which lines should carry bare integration momenta. All remaining momenta will be assigned automatically.

Sometimes it is necessary to use more sophisticated distributions. For example, the user may want to use his favorite momenta like $k-p_1, k-p_2, k-p_3$, etc. assigned to some definite lines (see e.g. [5]). For such cases, DIANA provides the possibility to define momenta only for the virtual lines, and the full set of topologies (obtained by exchanging external momenta) will be defined from these "internal" topologies attaching the external legs. In this case momenta for internal parts are defined in terms of combinations of loop momenta (k in the above example) and some "abstract" tokens (p_1, p_2 etc.), and for each topology DIANA expresses these tokens in terms of external momenta.

Another example: occasionally topologies are generated by more complicated ("generic") ones by scratching lines. In such cases one wants to stick to the momenta introduced for the lines which are kept [4][4]. E.g. the user investigates the generic topology

```
generictopology A =
 (-2,2)(-1,1)(1,3)(3,2)(2,4)(4,1)(3,4):
 p1+k2,p1+k1,k1,k2,k2-k1.
```

Then DIANA will generate topologies from this one by scratching lines in the following manner:



[4]For details look at http://www.physik.uni-bielefeld.de/~tentukov/generictopology.html

The topology name is constructed by DIANA as follows: the name of the "generic" topology is appended by the index of "scratched" lines separated by underscore "_".

To run an external command `cmd`, the function `\exec(cmd)` can be used. This function executes the command in the background without waiting for it to be completed. So, this function may be used to **paralyse** the evaluation of a process by running more then one FORM job simultaneously.

To avoid overloading the processor, by default at each time only one job is actually running while all the rest is waiting in the queue. On SMP[5] computers, the number of simultaneously running jobs can by changed by means of a command line option `-smp`. Thus, `-smp 8` tells DIANA to run on the computer 8 jobs simultaneously while all the rest is queued.

To synchronize the TM[6] program with all started jobs, the function `\waitall(timeout)` is used. It suspends execution of the TM program until all jobs are completed. Each "`timeout`" milliseconds the function reports the number of jobs which are not finished yet.

In case of a cluster of computers with a current directory shared by means of NFS[7] e.g., the function `\exec()` can use DIANA servers running on other computers. To run DIANA as a server, the user must use the command line option `-d #`, where `#` is the number of jobs which can run on this computer simultaneously. Thus, each computer on which the user has executed the command `diana -d 1 -q` is ready to perform the commands queued by the function `\exec()` (we assume that the current directory is shared by NFS among all computers in the cluster). Here `-d 1` tells DIANA to run a daemon[8] accepting only one connection, and the option `-q` terminates the father DIANA process. For SMP computers the optimal argument for the `-d` option would be the number of processors.

[5]Symmetric MultiProcessing
[6]TM is an abbreviation for "Text Manipulating", see [1].
[7]Network File System
[8]A program running in the background and listening to some port.

Two functions, `\exec()` and `\waitall()`, permit the user to organize a simple parallel session for evaluations on SMP and/or cluster of independent computers with shared disk space. But, very often this is not enough. Indeed, let us suppose that all results must be collected into one resulting file `log.all`, while every job produces a file `log.#` where `#` is the order number of the job. Since jobs are running and completing independently from each other, we can only collect all `log.#` into `log.all` after *all* the jobs are finished. This leads to producing a lot of files `log.#` at an intermediate step, which can overload a file system.

The simple solution is to allow some of newly started jobs to be synchronized with all previously started jobs. Indeed, in this case after each job we can start another "slave" job, which appends the `log.#` file containing the result of the "master" job to the file `log.all`. To obtain a correct order of the file `log.all`, this copying job must be performed only after all earlier jobs are finished.

Another problem with cluster computations is that the optimal placement of the resulting file `log.#` is usually a `/tmp` directory which is local against a current node[9], on which a job "number `#`" is performed. But to do this, the "slave" copying job has to "stick" to the "master" job, i.e. it must be performed on the same node as the "master" job.

In order to create such a "slave" job, the function `\stick(cmd)` must be used. This function is similar to `\exec(cmd)` with two exceptions. It performs the command `cmd` only after all earlier jobs are completed. This function in general is used to sum up all produced files. That is why it performs `cmd` on the same computer as the preceding job.

As an example, let us consider the following script `runf`:

```
#!diana -smp 1 -c runpar.tml
\STARTSERVERS(phya25,phya26,phya27,phya28,phya29)
\system(echo > log.all)
```

[9]DIANA assumes the "node" is the IP address of a server, so the conception of "nodes" is actually supported only for clusters. For SMP, the whole computer is assumed to be a single node.

```
\REPEAT(N)
\exec(form -d i=\get(N) do.frm > /tmp/log.\get(N))
\stick(cat /tmp/log.\get(N) >> log.all)
\stick(rm /tmp/log.\get(N))
\ENDREPEAT()

\waitall(2000)
```

The user enters: `runf 186 200` , and the system executes:
`diana -smp 1 -c runpar.tml runf 186 200.`

The file `runpar.tml` contains definitions of various TM functions and other settings; in particular, it redefines the comment character as `#`. The macro `\STARTSERVERS(list)` checks if each server is working and, if not, it starts the server by means of the `ssh`. For example, for the host phya26 the following command will be performed:

`ssh phya26 cd CD ; diana -d 1 -q`

where "`CD`" is a current directory, e.g. `/home/user/jobs`.

The operator `\system(cmd)` executes the command `cmd` synchronously, i.e. it waits for the command to be completed and returns an exit code. Here it is used to produce an empty file "`log.all`".

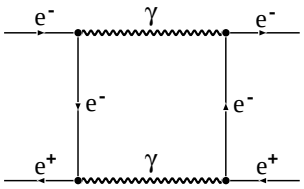
All the instructions between `\REPEAT(N)...\ENDREPEAT()` are cycled with `N=186,...,200`. We assume that there is some folder file, say, `tt.in` with FORM input for each diagram. The FORM program `do.frm` evaluates a diagram by virtue of including a fold from the folder `tt.in` via an instruction like `#include tt.in # n'i'`. The macro definition `i` comes from the command line `form -d i=\get(N) do.frm`, where `\get(N)` runs from 186 to 200. Each FORM job saves the result to the local directory, but the corresponding concatenation is performed by `\stick(cat ...)` on the same computer. At the end, all results will be collected in the file `log.all`, and all intermediate files `\tmp\log.#` will be removed.

After all jobs are queued, the function `\waitall(2000)` will report every 2 seconds how many jobs are not yet completed.

Recent applications of DIANA in our collaboration are e^+e^- annihilation into $t\bar{t}$ and Bhabha scattering.

The $t\bar{t}$ production has been calculated including hard bremsstrahlung and various comparisons with other groups have been performed. For further information about the results we refer to recent notes on this subject [6,7].

Concerning Bhabha scattering, since we are heading for two-loop calculations, we perform all calculations in arbitrary dimension $d = 4 - 2\varepsilon$. The algebraic part of this calculation has been presented in [8]. Here we discuss only the evaluation of one-loop master integrals, in particular of box diagrams. The method applied uses evaluation of difference equations [9]. We investigate now the following (scalar) box diagram with two (zero mass) photons in the s-channel:



The corresponding master integral in d dimensions can be represented as

$$
\begin{aligned}
&I_4^{(d)} = \\
&\frac{2\Gamma(2-\frac{d}{2})}{t-4m^2}\Bigg[\\
&\frac{(m^2)^{(\frac{d}{2}-2)}}{s}\int_0^1 \frac{dx\; x^{\frac{d-5}{2}}}{1-k_1x}\; {}_2F_1(1,\frac{d-3}{2},\frac{d-2}{2},k_2x) \\
&-\frac{\sqrt{\pi}(-s)^{(\frac{d}{2}-2)}}{m\sqrt{s}\;2^{(d-3)}}\;\frac{\Gamma\left(\frac{d-2}{2}\right)}{\Gamma\left(\frac{d-3}{2}\right)}\;\int_0^1 \frac{dx\; x^{\frac{d-5}{2}}}{1-k_3x}\;\frac{1}{\sqrt{1-k_4x}} \\
&-\frac{(m^2)^{(\frac{d}{2}-2)}}{s}\frac{\Gamma\left(\frac{d-2}{2}\right)}{\Gamma\left(\frac{d-3}{2}\right)}\;\sqrt{1-\frac{4m^2}{t}}\;\Bigg(\\
&(1-\frac{t}{4m^2})^{(\frac{d}{2}-2)}\sqrt{\pi}\;\Phi(k_5,1,\frac{d}{2}-2) \\
&-(1-\frac{4m^2}{t})^{(\frac{d}{2}-2)}\frac{1}{\sqrt{\pi}}\int_0^1 \frac{dx\; x^{\frac{d-5}{2}}}{\sqrt{1-x}}(1-k_6x)^{(2-\frac{d}{2})} \\
&\Phi(\frac{-k_5k_6x}{1-k_6x},1,\frac{d}{2}-2)\Bigg)\Bigg],
\end{aligned}
$$

where the $k_i (i = 1 \ldots 6)$ are the following kine-

matical variables

$$
\begin{aligned}
k_1 &= 1 - \frac{4m^2}{s} \quad , \\
k_2 &= -4m^2(\frac{1}{s} + \frac{1}{t-4m^2}) \equiv -m^2 z \\
k_3 &= 1 + \frac{s}{t-4m^2}, \quad k_4 = 1 - \frac{s}{4m^2} \\
k_5 &= 1 + \frac{t-4m^2}{s}, \quad k_6 = \frac{4m^2}{t}
\end{aligned}
$$

and $\Phi(z,1,a)$ is the Lerch function

$$\Phi(z,1,a) = \sum_{k=0}^{\infty} \frac{z^k}{k+a}$$

For $s > 4m^2$ and $t < 0$ it is easy to see that we remain in the analyticity domain of all occurring expressions, except for the factor $(-s)^{(\frac{d}{2}-2)}$. This result finally allows to expand the diagram in $\varepsilon = \frac{4-d}{2}$. In the two-loop calculation we need to expand up to order $O(\varepsilon)$, which will yield finite contributions when multiplied, e.g., by divergent counterterms, containing terms of order $O(\frac{1}{\varepsilon})$. Due to the overall factor $\Gamma(2-\frac{d}{2})$, coming from the infrared divergence of the diagram, the expansion of the various contributions in square brackets must be performed up to order $O(\varepsilon^2)$. We also see that the imaginary part of the finite part of the diagram comes from the above $(-s)^{(\frac{d}{2}-2)}$ when expanded up to order $O(\varepsilon)$.

Crossed diagrams can be obtained from this form as well if the result of the expansion yields analytic expressions (for which the analytic continuation should be known in general) or again the kinematical variables are such that a numerical integration hits no singularities.

Acknowledgement The authors wish to thank the organizers of 'RADCOR' and 'Loops and Legs 2002' for the nice organization of the conference. M.T. and O.V.T thank the DFG for financial support. J.F. and O.V.T want to thank DESY Zeuthen for travel support for visits.

REFERENCES

1. M. Tentyukov, J. Fleischer, Comput. Phys. Comm. 132 (2000) 124.
2. J. Fleischer *et al.*, Phys. Lett. **B459** (1999) 625; J. Fleischer, O. V. Tarasov and M. Tentyukov, Nucl. Phys. Proc. Suppl. **89** (2000) 112; F. Jegerlehner, M. Y. Kalmykov and O. Veretin, Nucl. Phys. **B641** (2002) 285; J. Fleischer *et al.*, hep-ph/0202109; A. Onishchenko and O. Veretin, hep-ph/0209010; M. Awramik *et al.*, hep-ph/0209084.
3. J. Fleischer, M. Tentyukov, "A Feynman diagram analyser DIANA: Graphic facilities", in: P. Bhat and M. Kasemann (eds.), Proc. of the VIIth Int. Workshop ACAT 2000, Batavia, USA, 16-20 Oct. 2000, AIP Conference Proceedings (Melville, New York, 2001), vol. 583, p. 193, hep-ph/0012189.
4. M. Tentyukov, J. Fleischer, Contribution to ACAT 2002, Moscow, Russia, June 24-28, 2002, hep-ph/0210179.
5. J. Fleischer, F. Jegerlehner, O.V. Tarasov, Nucl. Phys. **B566** (2000) 423.
6. J. Fleischer, T. Hahn, W. Hollik, T. Riemann, C. Schappacher and A. Werthenbach, 'Complete electroweak one-loop radiative corrections to top-pair production at TESLA - a comparison', hep-ph/0202109.
7. J. Fleischer, J. Fujimoto, T. Ishikawa, A. Leike, T. Riemann, Y. Shimizu and A. Werthenbach, 'One-loop corrections to the process $e^+e^- \to t\bar{t}$ including hard bremsstrahlung, hep-ph/0203220.
8. J.Fleischer, T. Riemann, O.V. Tarasov and A. Werthenbach, 'Massive two-loop Bhabha scattering - the factorizable subset', these proceedings; see also 'Factorizing one-loop contributions to two-loop Bhabha scattering and automatization of Feynman diagram calculations', Contribution to ACAT 2002, Moscow, Russia, June 24-28, 2002, hep-ph/0203220.
9. J.Fleischer, F.Jegerlehner and O.V. Tarasov, 'A new hypergeometric representation of one-loop scalar integrals', in preparation. See also O.V. Tarasov, Nucl. Phys. Proc. Suppl. **89** (2000) 237;